Process (ETL Project)

- Created Python virtual environment
    - Inside the project folder
- Activate venv
    - By default it is already activated (VSCode)
- pip install apache-airflow
- Created kafka_stream.py file
    - Call api and fetch the data with a proper format
- Create docker-compose.yml file
    - To keep everything in the Docker container
    - It will keep checks on all our installations so we can run it on other machines

1. Install Dependencies

Before installing services, ensure you have the necessary system packages:

- sudo apt update && sudo apt upgrade -y
- sudo apt install -y openjdk-11-jdk python3 python3-pip wget curl netcat

2. Install and Configure Services Manually

You'll need to install and start the following services:

A. Install and Start Zookeeper

Install:

- sudo apt install -y zookeeper zookeeperd

Start Zookeeper:

- sudo systemctl start zookeeper
- sudo systemctl enable zookeeper

Check if Zookeeper is Running:

- echo "ruok" | nc localhost 2181
    - If it responds with imok, Zookeeper is running successfully.

B. Install and Start Kafka

Download Kafka:

- wget https://downloads.apache.org/kafka/3.6.0/kafka_2.13-3.6.0.tgz
- tar -xvzf kafka_2.13-3.6.0.tgz
- mv kafka_2.13-3.6.0 kafka
- cd kafka

Start Kafka Server:

- bin/kafka-server-start.sh config/server.properties

To test Kafka, create a topic:

- bin/kafka-topics.sh --create --topic test_topic --bootstrap-server localhost:9092
- bin/kafka-topics.sh --list --bootstrap-server localhost:9092

C. Install and Start Schema Registry

Download Confluent Schema Registry:

- wget https://packages.confluent.io/archive/7.4/confluent-7.4.0.tar.gz
- tar -xvzf confluent-7.4.0.tar.gz
- mv confluent-7.4.0 confluent
- cd confluent

Start Schema Registry:

- bin/schema-registry-start etc/schema-registry/schema-registry.properties

D. Install Apache Airflow

- pip3 install apache-airflow
- export AIRFLOW_HOME=~/airflow
- airflow db init

Start the webserver:

- airflow webserver --port 8080

Start the scheduler:

- airflow scheduler

E. Install and Start PostgreSQL

- sudo apt install -y postgresql postgresql-contrib
- sudo systemctl start postgresql
- sudo systemctl enable postgresql

Create Database and User:

- sudo -u postgres psql

Inside PostgreSQL shell, run:

- CREATE DATABASE airflow;
- CREATE USER airflow WITH ENCRYPTED PASSWORD 'airflow';
- GRANT ALL PRIVILEGES ON DATABASE airflow TO airflow;
- \q

F. Install Apache Spark

Download Spark:

- wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
- tar -xvzf spark-3.5.0-bin-hadoop3.tgz
- mv spark-3.5.0-bin-hadoop3 spark

Start Spark Master:

- ./spark/sbin/start-master.sh

Start Spark Worker:

- ./spark/sbin/start-worker.sh spark://localhost:7077

G. Install and Start Cassandra

Install Cassandra:

- wget https://downloads.apache.org/cassandra/4.1.3/apache-cassandra-4.1.3-bin.tar.gz
- tar -xvzf apache-cassandra-4.1.3-bin.tar.gz
- mv apache-cassandra-4.1.3 cassandra

Start Cassandra:

- ./cassandra/bin/cassandra -R

To verify:

- ./cassandra/bin/cqlsh
    - If it enters the Cassandra shell, it's working fine.

3. Configure Services to Work Together

Each service must be properly configured to interact with others.

- **Kafka should connect to Zookeeper (localhost:2181)**
- **Schema Registry should connect to Kafka (localhost:9092)**
- **Airflow should use PostgreSQL (localhost:5432)**
- **Spark should be able to fetch data from Kafka**
- **Processed data should be stored in Cassandra (localhost:9042)**

---

4. Run Your Pipeline

Now that everything is installed and configured:

1. Start Zookeeper
2. Start Kafka
3. Start Schema Registry
4. Start Airflow (Webserver & Scheduler)
5. Start PostgreSQL
6. Start Spark Master & Worker
7. Start Cassandra

Then run your DAGs and Kafka consumers to process and store data.

If want to keep it all inside Docker for easy installations:

```
version: '3'

services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.4.0
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    healthcheck:
      test: ['CMD', 'bash', '-c', "echo 'ruok' | nc localhost 2181"]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      - confluent

  broker:
    image: confluentinc/cp-server:7.4.0
    hostname: broker
    container_name: broker
    depends_on:
      zookeeper:
        condition: service_healthy
    ports:
      - "9092:9092"
      - "9101:9101"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS:
PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_METRIC_REPORTERS: io.confluent.metrics.reporter.ConfluentMetricsReporter
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_CONFLUENT_LICENSE_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_CONFLUENT_BALANCER_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
```

```
    KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
    KAFKA_JMX_PORT: 9101
    KAFKA_JMX_HOSTNAME: localhost
    KAFKA_CONFLUENT_SCHEMA_REGISTRY_URL: http://schema-registry:8081
    CONFLUENT_METRICS_REPORTER_BOOTSTRAP_SERVERS: broker:29092
    CONFLUENT_METRICS_REPORTER_TOPIC_REPLICAS: 1
    CONFLUENT_METRICS_ENABLE: 'false'
    CONFLUENT_SUPPORT_CUSTOMER_ID: 'anonymous'
  networks:
    - confluent
  healthcheck:
    test: [ "CMD", "bash", "-c", 'nc -z localhost 9092' ]
    interval: 10s
    timeout: 5s
    retries: 5

schema-registry:
  image: confluentinc/cp-schema-registry:7.4.0
  hostname: schema-registry
  container_name: schema-registry
  depends_on:
    broker:
      condition: service_healthy
  ports:
    - "8081:8081"
  environment:
    SCHEMA_REGISTRY_HOST_NAME: schema-registry
    SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: 'broker:29092'
    SCHEMA_REGISTRY_LISTENERS: http://0.0.0.0:8081
  networks:
    - confluent
  healthcheck:
    test: [ "CMD", "curl", "-f", "http://localhost:8081/" ]
    interval: 30s
    timeout: 10s
    retries: 5
control-center:
  image: confluentinc/cp-enterprise-control-center:7.4.0
  hostname: control-center
  container_name: control-center
  depends_on:
    broker:
      condition: service_healthy
    schema-registry:
      condition: service_healthy
  ports:
    - "9021:9021"
  environment:
```

```
      CONTROL_CENTER_BOOTSTRAP_SERVERS: 'broker:29092'
      CONTROL_CENTER_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
      CONTROL_CENTER_REPLICATION_FACTOR: 1
      CONTROL_CENTER_INTERNAL_TOPICS_PARTITIONS: 1
      CONTROL_CENTER_MONITORING_INTERCEPTOR_TOPIC_PARTITIONS: 1
      CONFLUENT_METRICS_TOPIC_REPLICATION: 1
      CONFLIENT_METRICS_ENABLE: 'false'
      PORT: 9021
    networks:
      - confluent
    healthcheck:
      test: [ "CMD", "curl", "-f", "http://localhost:9021/health" ]
      interval: 30s
      timeout: 10s
      retries: 5

  webserver:
    image: apache/airflow:2.6.0-python3.9
    command: webserver
    entrypoint: ['/opt/airflow/script/entrypoint.sh']
    depends_on:
      - postgres
    environment:
      - LOAD_EX=n
      - EXECUTOR=Sequential
      -
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@
postgres:5432/airflow
      - AIRFLOW_WEBSERVER_SECRET_KEY=this_is_a_very_secured_key
    logging:
      options:
        max-size: 10m
        max-file: "3"
    volumes:
      - ./dags:/opt/airflow/dags
      - ./script/entrypoint.sh:/opt/airflow/script/entrypoint.sh
      - ./requirements.txt:/opt/airflow/requirements.txt
    ports:
      - "8080:8080"
    healthcheck:
      test: ['CMD-SHELL', "[ -f /opt/airflow/airflow-webserver.pid ]"]
      interval: 30s
      timeout: 30s
      retries: 3
    networks:
      - confluent

  scheduler:
```

```
    image: apache/airflow:2.6.0-python3.9
    depends_on:
      webserver:
        condition: service_healthy
    volumes:
      - ./dags:/opt/airflow/dags
      - ./script/entrypoint.sh:/opt/airflow/script/entrypoint.sh
      - ./requirements.txt:/opt/airflow/requirements.txt
    environment:
      - LOAD_EX=n
      - EXECUTOR=Sequential
      -
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@
postgres:5432/airflow
      - AIRFLOW_WEBSERVER_SECRET_KEY=this_is_a_very_secured_key
    command: bash -c "pip install -r ./requirements.txt && airflow db upgrade && airflow
scheduler"
    networks:
      - confluent

  postgres:
    image: postgres:14.0
    environment:
      - POSTGRES_USER=airflow
      - POSTGRES_PASSWORD=airflow
      - POSTGRES_DB=airflow
    logging:
      options:
        max-size: 10m
        max-file: "3"
    networks:
      - confluent

  spark-master:
    image: bitnami/spark:latest
    command: bin/spark-class org.apache.spark.deploy.master.Master
    ports:
      - "9090:8080"
      - "7077:7077"
    networks:
      - confluent
  spark-worker:
    image: bitnami/spark:latest
    command: bin/spark-class org.apache.spark.deploy.worker.Worker
spark://spark-master:7077
    depends_on:
      - spark-master
    environment:
```

```
      SPARK_MODE: worker
      SPARK_WORKER_CORES: 2
      SPARK_WORKER_MEMORY: 1g
      SPARK_MASTER_URL: spark://spark-master:7077
    networks:
      - confluent

  cassandra_db:
    image: cassandra:latest
    container_name: cassandra
    hostname: cassandra
    ports:
      - "9042:9042"
    environment:
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
      - CASSANDRA_USERNAME=cassandra
      - CASSANDRA_PASSWORD=cassandra
    networks:
      - confluent
networks:
  confluent:
```